Engineering the Virtual Node Layer for Reactive MANET Routing

***Abstract*—The VNLayer approach [1] simplifies software development for MANET by providing the developers an abstraction of a network divided into fixed geographical regions, each containing a virtual server for network services.**

In this paper, we present our study on reactive MANET routing over the VNLayer. During this research, we identified in our initial VNLayer implementation three major limitations that lead to heavy control traffic, long forwarding paths and frequent message collisions in MANET routing. To address the problems, we changed the assumptions made by the VNLayer on the link layer and extended the operations allowed by VNLayer. This results in a VNLayer implementation that can be tuned to optimize the performance of traffic intensive applications (such as routing) while maintaining their simplicity and robustness.

Simulation results showed that VNAODV, a VNLayer based routing protocol adapted from AODV [4], delivers more packets, generates less routing traffic and creates more stable routes than AODV in a dense MANET with high node motion rates. This research validated that the VNLayer approach makes software development for MANET easier and improves the performance of MANET protocols.

Keywords: MANET, Routing, VNLayer, Virtual Nodes, AODV

I. OVERVIEW

A. The VNLayer Approach

The Virtual Node Layer (VNLayer) [1] is a cluster-based approach [5] to Mobile Ad-hoc Networks (MANET). The VNLayer approach divides a network into regions at fixed geographical locations. Within each region, a subset of physical nodes cooperates to emulate a virtual server (***virtual node***) that can be addressed from a client's application layer as if it was an actual physical, static, server device. The emulation can be implemented in different ways. A common approach, studied in [1,2,3], is to have physical nodes in each region elect a leader. By a state synchronization protocol, non-leaders maintain replicated states which are consistent with the leader's state. Hence, the virtual node in a region can maintain persistent state and be fault tolerant even when individual physical nodes might fail or leave a region. Furthermore, the fixed virtual nodes also create a level of hierarchy in a flat MANET, allowing MANET protocols to operate more efficiently and reliably.

As a generalized programming abstraction, the VNLayer lies between the MANET link layer and applications built upon it (the ***application layer***). The VNLayer hides complexities such as location checking, leader election, packet buffering and state replication from programmers. Because programmers only need to deal with the VNLayer, rather than dealing with a set of highly unpredictable physical nodes and the unreliable wireless channel, they can deploy applications on both mobile devices and static virtual servers with greater ease and efficiency.

B. Why Implementing AODV over the VNLayer?

Research on a VNLayer based MANET address allocation protocol [3] proved that the VNLayer approach is practical for simple protocols with little overhead, but can protocols involving continuous activity and generating significant overhead also be supported efficiently with the VNLayer approach? One rigorous test of this would be adaptation of a mature MANET routing protocol to the VNLayer approach. Can the adapted routing protocol deliver a packet, in the absence of message losses, with a bounded delay whenever there is a viable forwarding path? To answer this question, we created VNAODV, an adapted version of AODV [4], a popular MANET routing protocol.

The main strength of AODV is its reactive nature. Since route entries are created only when they are needed, there is no need for periodic routing updates. In addition, AODV's use of route sequence numbers ensures route freshness and prevents routing loops.

The main weakness of AODV is its use of flooding in route requests. A route discovery can involve every single physical node in a MANET. This makes AODV's routing overhead proportional to the total number of physical nodes in a network. In a dense MANET, each route discovery can trigger a round of broadcast storm. The use of expanding ring search and local route repair [4] only reduces the severity of the problem.

Another problem of AODV is that it picks routes based on route length and route freshness, without considering the stability of a route. A router with the shortest route is picked as the next hop. However, it can move away very soon and causing the route to fail. When the motion rates of physical nodes in a MANET are high, the routes created by AODV fail frequently, leading to heavy routing traffic.

A cluster-based scheme like the VNLayer approach is the natural solution to the two problems above. With AODV implemented over the VNLayer, routing is done by virtual nodes. In a region, only the leader, rather than every physical node, sends and forwards routing messages. Compared with the connections among physical nodes that are moving, the connections and topology among the fixed virtual nodes are much more stable. Therefore, compared with AODV, VNAODV is expected to generate less routing overhead and to create more stable routes.

The goal of this study is to find out whether VNAODV can perform better than AODV. With our initial implementation of the VNLayer (referred to in this paper as ***the basic VNLayer implementation***), we found that VNAODV generates heavy control traffic overhead, creates forwarding paths much longer than the ones created by AODV and drops a lot of messages. In this paper, we present how we re-engineered the VNLayer implementation to provide better service to traffic intensive applications such as MANET routing.

C. Related Works

The VNLayer approach is essentially a clustering [5] scheme in which physical nodes are grouped to create a level of hierarchy in the originally flat MANET. There have been many works done on using clustering for MANET routing. Cluster Overlay Broadcast (COB) [6] is similar to AODV [4], but with route request and response messages (RREQs and RREPs) relayed only by cluster heads. When a cluster head receives an RREP message, it marks itself as active for a specific session. However, the route created by COB can only be used once. Another cluster based routing protocol is CEDAR [7]. CEDAR uses a degree based core extraction algorithm to elect cluster heads (core nodes) and to form dominating sets among physical nodes in a MANET. In CEDAR, to reduce the negative impact of flooding, the route request messages are flooded by core nodes by unicast, rather than by local broadcast.

Compared with other cluster based routing schemes, the VNLayer approach has the following advantages. First, as a generalized programming abstraction, the VNLayer can be used by different applications, not just routing applications. Different applications built over the VNLayer can share the capabilities provided by it. Second, the VNLayer can maintain persistent application state among emulator nodes in the face of leadership changes. Third, with the fixed region setting of the VNLayer approach, membership changes taking place in one region won't cause other regions to re-cluster. Therefore, the VNLayer approach doesn't have the rippling effect [5].

One major assumption for a VNLayer based system is that physical nodes in a MANET can determine their current geographical location. In this paper, we assume each physical node is equipped with GPS. However, if it is too expensive to do, the majority of physical nodes can infer their locations from the location of a small subset of GPS equipped physical nodes, using a localization algorithm [8].

D. Structure of the Paper

The paper is structured as follows. Section II first presents our initial VNLayer implementation and its limitations with routing applications. Then we discuss our extensions to the initial implementation and their impacts on the VNLayer guarantees. Section III first describes the operations of VNAODV without the VNLayer extensions. It then explains how the VNLayer extensions are used to improve VNAODV's performance. Section IV presents the performance evaluation results. Section V gives our conclusions and future works.

II. ENGINEERING THE VNLAYER

A. The Basic VNLayer Implementation

The basic VNLayer implementation was based on M. Spindel's work in [2]. In the implementation, a MANET is tiled with equal sized square regions. A region can have up to 8 ***neighbor regions*** around it. Each region is set to a size such that a message sent by any physical node can reach all the physical nodes in the sender's region and its neighbor regions. All physical nodes are assumed to have the same radio range, to know their geographical locations and the region setting. Because a physical node always knows the region it is in, region based clusters can be formed easily.

In each region, a subset of physical nodes emulates a virtual node. Each physical node emulating a virtual node is called an ***emulator***. In each region, one emulator takes the ***leader*** role and the others take ***non-leader*** roles. When an emulator becomes a non-leader, it synchronizes its *entire* application state with the leader of the region.

When a physical node arrives in an empty region, it restarts and initializes the virtual node of the region. Since each non-empty region will have a leader, the virtual node in a non-empty region is said to be ***up***. The virtual node in an empty region is said to have ***failed***. Each physical node may host a client process. A physical node hosting only a client process is called a ***pure client***. In our simulations, every physical node is an emulator and hosts a client process.

In each region, emulators run the same application server program and process incoming application messages the same way. However, only the leader sends out response messages. The other emulators, the non-leaders, buffer their response messages in a ***sending queue***. To allow all emulators in a region to hear a message destined for the region, at the link layer, messages are always sent by ***local broadcast*** (with a broadcast destination address). To allow all emulators in a region to receive messages in the same order, incoming application message are buffered and sorted by their sending time before they are passed to the application layer.

When a non-leader's application state is consistent with its leader's state, it produces the same set of response messages its leader does. For *every* application message received from its leader, a non-leader checks its sending queue for a match. If a matched message can be found, the non-leader removes it from its sending queue. Otherwise, the non-leader synchronizes its *entire* application state with the leader. In a region, when the leader leaves or crashes, a non-leader is elected as the new leader. The new leader then sends out the messages in its sending buffer. Non-leaders hence work as backup servers.

Inter-region messages are always relayed by virtual nodes. This means a client process can communicate only with its local virtual node. A virtual node can communicate only with virtual nodes in its neighbor regions. This guarantees the reliability of inter virtual node communications. It also guarantees that in each region, either all emulators receive an application message or none do (the ***atomicity property***).

B. Problems with Routing Applications

A virtual node providing routing service is called a ***vrouter***. Fig. 1 shows how a message is forwarded by vrouters using the basic VNLayer implementation. When the client process on pure client node 3 generates a message destined for node 7, the VNLayer on the node sends the message to the local region so that all the emulator nodes can hear it. The message is then processed by the local vrouter (emulated by node 1 and 2). Leader node 1 forwards the message to region 2.0, while non-leader node 2 buffers an identical message in its sending queue. When node 2 hears the message sent by node 1, it removes the matching message from its sending queue. The vrouters in region 2.0, 3.1 and 2.2 then forward the packet all the way to the destination node 7, with node 4, 5 and 6 as region leaders and node 7 and 8 as backup routers. Node 9 is another pure client node and is not involved in the routing at all.

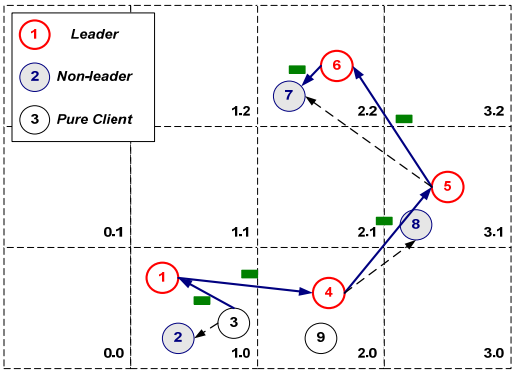

Figure 1. MANET Routing using the Basic VNLayer Implementation

For routing applications, we found the following three problems with the basic VNLayer implementation:

1. The implementation requires the entire application state to be synchronized among emulators. It also requires a non-leader to use every incoming application message from its leader to detect state inconsistencies. When a MANET is dense, dynamic and the data traffic is heavy, the operations of a vrouter can involve a large application state (routing table) and frequent exchanges of routing and data messages. Many messages can be lost due to channel collisions. Since any application message (including the ones that don't affect the application state) missed by a non-leader may lead to a state synchronization, frequent message losses can cause heavy state synchronization traffic and lead to even more message losses.

2. The implementation requires a client process to communicate only with its local vrouter. It also requires a vrouter to communicate only with vrouters in its neighbor regions. With the region sizes used by the implementation, the average distance between two vrouters is less than half of the radio ranges of physical nodes. These rules hence make the forwarding paths created by vrouters much longer than the ones created by non-clustering routing protocols that use physical nodes as routers. When the data traffic is heavy, longer forwarding paths translates to heavy data forwarding traffic and frequent data delivery failures.

3. In the implementation, every application message is sent by local broadcast. Messages sent by local broadcast collide with other messages frequently and are susceptible to transmission failures. It has been shown that local broadcast based flooding of route requests is very harmful [7]. When the data traffic is heavy, the excessive use of local broadcast causes frequent message losses and hurts the routing performance badly.

C. The Extended VNLayer Implementation

To solve the problems identified above, we changed the assumption made by the basic implementation on the link layer and extended its implementation choices. These changes result in an ***extended VNLayer implementation***.

1) Assumption on the Link Layer

The basic VNLayer implementation assumes a link layer without message loss. It sends all messages by local broadcast without worrying about its impact on the reliability of data transmissions. In the absence of message losses, the basic VNLayer implementation guarantees that a virtual node maintains its current application state as long as it doesn't fail.

The assumption made by the basic implementation on the link layer is clearly not realistic. Therefore, the extended VNLayer implementation assumes an 802.11 like link layer that is subject to message losses caused by collisions and radio interferences. Due to the existence of message losses, this implementation doesn't guarantee that a virtual node retains its current state even if the virtual node never fails. Due to state inconsistencies among emulators, the application state on a virtual node may have occasional arbitrary changes when the leadership of its region changes.

2) Extensions to State Synchronization Mechanism

Selective State Synchronization: As an alternative to having the entire application state synchronized, applications built over the VNLayer are given the option to have only the critical part of the state (the *hard state*) synchronized. This way, the size of state synchronization messages can be reduced. ***Hard state*** is defined as the part of application state that is critical to the correct operation of an application. The rest part of application state is considered ***soft state***. To use this option, at the application layer, a programmer needs to determine which part of the application state is hard state.

Selective State Consistency Check: As an alternative to using every leader application message to check for state inconsistencies, a non-leader is given the option to detect state inconsistencies using only messages that are more relevant to the application state. In the VNLayer message header, a "subtype" field is used to specify a message's relevance to the application state. For example, messages originated from virtual nodes are tagged as ***Server Messages***. Messages relayed by virtual nodes are marked as ***Forwarded Server Messages***. A non-leader can choose not to check a subtype of messages. This way, the number of unnecessary state synchronizations can be reduced. To use this option, an application needs to classify outgoing messages into different subtypes.

It needs to be noted that the two options reduce the state synchronization overhead at the price of weakened guarantee on state consistency among emulators.

3) Extensions to Communication Rules

In order to shorten the forwarding paths created by VNAODV, we relaxed the communication rules in the basic implementation. Now, a client process is given the option, called ***Direct Receipt (DR)***, to accept messages received from virtual nodes in its neighbor regions. A virtual node is given the option, called ***Long Links (LL)***, to accept messages received from any other virtual node or client process. Clearly, DR and LL can be used to allow more virtual nodes and client processes to communicate with each other directly. However, the shortened forwarding paths come at the cost of reduced reliability of data transmission between virtual nodes. When a virtual node is allowed to talk to a non-neighboring virtual node, we can no longer guarantee the *atomicity property* on virtual nodes, even when there is no message loss.

4) Extensions to Data Transmission Methods

In the link layer, for a message destined for a region to be heard by all emulator nodes in the region, in addition to using

local broadcast, we have another option, ***Directed Broadcast (DB)***.That is, a message is sent by unicast to the address of a physical node in the target region. To use DB, every physical node to use the promiscuous mode on their NICs so that they can hear a unicast message not destined for it.

For a virtual node to send a message to a client process, using DB is easy because the address of the physical node hosting the client process is known. To allow virtual nodes to communicate with each other using DB, the VNLayer is implemented to keep track of the addresses of region leaders using messages heard from them. When the VNLayer knows the address of the leader of an outgoing message's next hop region, it sends the message by DB to the leader's address. Otherwise, local broadcast is used.

With local broadcast, a sender can't determine whether a message has been received by a potential recipient. With DB, a data transmission is more reliable. Before sending, the sender can determine if the recipient is able to receive a message. After sending, it can find out if a message was received. (In 802.11, when unicast is used, ARP, CMSA/CA, data acknowledgement and retransmission can be used by the link layer to prevent, to detect and to recover from data transmission failures.) However, the improved reliability brought by DB comes at the cost of a more complicated VNLayer implementation and requiring all physical nodes to use the promiscuous mode.

III. VNLayer based Ad-hoc On Demand Routing

We now present how VNAODV works over the extended VNLayer implementation. We start with the operations of VNAODV when the extensions we added to the VNLayer implementation are not used. Then we explain how the VNLayer extensions improve the performance of VNAODV.

A. VNAODV without VNLayer Extensions

VNAODV is implemented based on the AODV package provided by the ns-2 simulator [9]. The core AODV algorithm and the setting of most parameters remained unchanged. To run over the VNLayer, the AODV implementation is modified to change routing entities from physical nodes to vrouters. Modifications are also done to provide the VNLayer API functions that handle and send application messages; initialize, retrieve and synchronize the state of vrouters. When the VNLayer extensions are not used, VNAODV operates as follows. Here, all messages are sent by local broadcast.

1) Route Discovery

When a vrouter receives a data message (DMSG) from a client process in its region, if it has a route for the DMSG, it sends the DMSG to the next hop vrouter or the destination. Otherwise, the vrouter saves the DMSG in a buffer (called ***RecvQueue***) and start a route discovery by flooding a request message, called an RREQ, to the network. The first hop vrouter, called the ***initiator***, puts its region id, a Reverse Route Sequence Number (RRSN) and a BCAST-id in the RREQ message. As in AODV, the RRSN is used by other vrouters to create reverse routes to the initiator. The BCAST-id is used by vrouters to avoid relaying the RREQ message more than once. When the RREQ message is received by the destination or a vrouter that has a fresh route for the destination, a response message, called an RREP, is sent back to the initiator (rather the client process that sent the DMSG).

2) Data Message Forwarding

Vrouters forward a DMSG region by region toward the destination. In the VNLayer header added to a DMSG, a field specifies the next hop vrouter. Each time a vrouter forwards a DMSG, it extends the lifetime of the route entry used by the DMSG. This way, a route in use doesn't expire unless a broken link is detected.

3) Route Maintenance

DMSGs are most frequently lost because a link fails for some reason (e.g., immediately after the leader node has left a region). Detecting link failures quickly is crucial to reducing DMSG delivery failures. Because AODV uses unicast to transmit DMSGs at every hop, it allows detection of link failure by any available means. For example, a failure can be detected when address resolution can't resolve the MAC address of the next hop router, RTS/CTS mechanism can't reserve the channel with the next hop, or no acknowledgement for the DMSG can be received and retransmission attempts also failed. This is called ***link layer detection***.

In this version of VNAODV, because local broadcast is used to send DMSGs, link layer detection can't be used. To solve this problem, we use passive DMSG acknowledgements, as suggested by the AODV specification [4].

Passive DMSG Acknowledgement: In VNAODV, each time a route entry is used to forward a DMSG, we set its lifetime to 3 times the maximum estimated one hop Round Trip Time (RTT) and mark it as "unacked". Before an "unacked" route entry expires, it can be set back to "acked" by any DMSG overheard from the next hop vrouter, with its lifetime set back to the regular route lifetime (10 seconds). Otherwise, the link to the next hop vrouter is considered unreliable and the route entry will be disabled or checked.

On all forwarding hops except the last hop, a DMSG is acknowledged by the forwarded DMSG from the downstream vrouter. At the last hop, the destination node confirms a DMSG with an explicit acknowledgement message to prevent the route used by the last hop from expiration.

Local Connectivity Check before Local Repair: When AODV detects a link failure through link layer detection, the link is most likely broken. Hence, it is reasonable to report the error right away or start a local repair by flooding an RREQ message. In the second option, the RREQ message carries an incremented route sequence number for the destination. Hence, a local route repair is a network-wide route discovery that expects a new route with a greater sequence number.

In VNAODV, because local broadcast is used, a possible link failure is always detected due to the absence of an acknowledgement (passive or explicit) to a DMSG. This can be caused by a message collision on either the DMSG or its acknowledgement. When a link failure is detected, it is likely that the link is still good.

To avoid excessive route discoveries, a Local Connectivity Check (LCC) is done to double check a link suspected of broken. To do an LCC, a vrouter marks the route entry

involved as "route in repair", buffers incoming DMSG messages using the route entry in the RecvQueue and broadcasts an RREQ message (with TTL=1) to the neighborhood. Because an LCC is not meant to find a fresher route, the RREQ message carries the current route sequence number the vrouter has for the destination. If the next hop vrouter is still reachable, it responds to the message with an RREP message. Receiving this message, the router set the status of the route entry back to "up" and delivers the DMSGs buffered. If no such message can be heard within 2 RTTs, the link is considered broken. In this case, the vrouter either reports the error upstream or does a local route repair.

As we can see, the use of local broadcast makes route maintenance in VNAODV more complicated than in AODV.

B. *VNAODV with VNLayer extensions*

In this section, we discuss how the VNLayer extensions improve the performance of VNAODV.

1) *On Reducing the State Synchronization Traffic*

To reduce VNAODV's state synchronization traffic, we used the following two VNLayer extensions:

Selective State Synchronization: With this extension, we synchronize only VNAODV's hard state to reduce the sizes of state synchronization messages. To use this extension, we need to determine which part of VNAODV's application state is hard state. In VNAODV's routing table, the largest part of its application state, there are route entries with different status (e.g., up, down, in-repair). Each route entry has many fields. The correctness of some parts of the routing table (e.g., the precursor list recorded by a route entry) only affects the performance of, rather than the correctness of routing.

In VNAODV's routing table, we consider only route entries that are up hard state. For each route entry that is up, fields such as destination id, hop count, route sequence number and next hop vrouter id are considered hard state. In addition, a vrouter's current reverse route sequence number and BCAST-id are also considered hard state. The physical node setting these two values for a vrouter can change due to node mobility.

Selective State Consistency Checks: To use this extension, at the application layer, VNAODV needs to classify application messages into subtypes so that they can be treated differently by the VNLayer. Routing messages originated from a vrouter, such as the RREQ, RREP and RERR messages, are marked by VNAODV as Server Messages. Forwarded DMSGs, RREQ, RREP and RERR messages, which compose the majority of VNAODV messages, are marked as Forwarded Server Messages. In VNAODV, to reduce the total number of state synchronizations, the VNLayer is set to use only Server Messages to check for state inconsistencies because they are considered more relevant to the application state.

2) *On Shortening Forwarding Paths*

To shorten VNAODV's forwarding paths, we used two extensions, Direct Receipt and Long Links.

Direct Receipt (DR): When a vrouter sends out a DMSG to its final destination in a neighbor region, the destination physical node can receive the message without the help of its local vrouter. At the VNLayer, we used the DR option to allow a client process to accept a message that is sent by its local vrouter or a vrouters in one of its neighbor regions. This way, a vrouter can deliver a DMSG directly to its final destination if the destination is in a neighbor region.

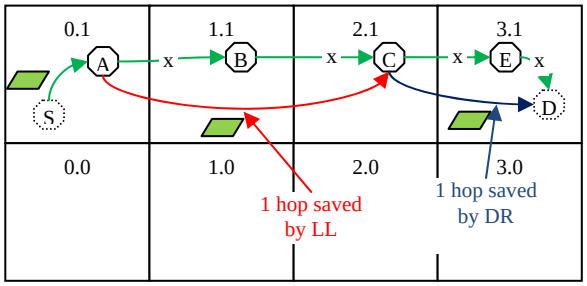

Figure 2. Route shortened by the DR and LL option

As illustrated in Fig. 2, with DR, the vrouter in region 2.1 directly delivers a DMSG to its destination D. To make DR work, at the VNLayer, we disallow a virtual node to accept a message sent to a client process in its region from another virtual node. Hence, the vrouter in region 3.1 doesn't do anything to the DMSG. By skipping the vrouter in the destination node's region, DR can shorten a forwarding path created by VNAODV by 1 hop.

Long Links (LL): Using the LL option at the VNLayer, a virtual node is allowed to accept any message it hears. Therefore, a vrouter can use any incoming RREP message it hears to update its routing table and can pick any vrouter within its radio range as a next hop. Fig. 2 shows that when LL is used, because the vrouter in region 0.1 can reach the vrouter in region 2.1 directly, the vrouter in region 1.1 is skipped. The forwarding path is shortened by another hop.

However, with LL used, it is possible that only a subset of emulators in a region can hear the messages sent by a vrouter to the region. In a vrouter, when the leader switches to an emulator that is out of the radio range of another vrouter, the link between the two vrouters will break. When this happens, local route repairs or network-wide route discoveries will be needed to fix routes using the link.

3) *On Improving the Reliability of Data Transmission*

To improve the reliability of data transmissions in VNAODV, at the VNLayer, application messages are sent by Directed Broadcast when the next hop is the final destination or when the address of the next hop region's leader is known. Otherwise, local broadcast is used, together with passive DMSG acknowledgements and LCC for route maintenance.

With this VNLayer extension used, for continuous DMSG transmissions to a vrouter, passive DMSG acknowledgements and LCC are rarely needed. This is because when local broadcast has to be used to send a DMSG to a vrouter, using one passive acknowledgement, the sender can acquire the address of the next hop region's leader. DB and link layer detection can be used for DMSG transmissions that follow, until the leader of the vrouter changes. In addition, when a DMSG is sent to its final destination, because DB can always be used, explicit DMSG acknowledgement is not needed at all. Therefore, the use of DB makes data transmissions more reliable and route maintenance much simpler.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the effect of the VNLayer extensions on VNAODV's performance, in terms of Packet Delivery Fraction ***(PDF)***, forwarding path length and routing traffic overhead. We simulated VNAODV with an ns-2 based VNLayer simulator, VNSim [3]. In our simulations, a 700m ×700m network is divided into 64 87.5m×87.5m square regions. The radio range of each physical node is set to 250m. The network contains 60 to 240 physical nodes. The 802.11 channel bandwidth is set to 11Mbps, with RTS-CTS disabled.

Node mobility patterns are generated with CanuMobiSim-1.3.4 using the Random Waypoint Model. Two motion modes, slow mode and fast mode are used. For both modes, the average pause time is set to 100 seconds. For the slow motion mode, the average motion rate is set to 1.825m/s (walking speed). For the fast motion mode, the average motion rate is set to 17.52m/s (vehicle speed).

Various numbers of Constant Bit Rate (CBR) sessions are created between random pairs of physical nodes. No two CBR sessions share either the source or the destination. Each session is set to transmit ten 64 byte UDP messages per second and to last till the end of a simulation. Each simulation lasts 450 seconds. The trace for the first 50 seconds in each simulation is skipped in our measurements to allow routing to stabilize. We repeated each simulation at least 5 times for each data point. Error bars are generated with confidence level of 95%.

A. *Effect of Selective State Synchronization and Selective State Consistency Checks*

We first study the impact of selective state synchronization and selective state consistency checks on the performance of VNAODV. Fig. 3 shows the PDF of AODV and VNAODV (with or without the two extensions) with different number of CBR sessions. Here, the Direct Receipt extension is already used by VNAODV to skip the vrouter in the region of the destination. The plot shows that when the two extensions are not used, VNAODV's PDF drops fast with increasing number of CBR sessions, due to the large state synchronization messages and frequent state synchronizations. AODV's PDF scales better than VNAODV due to its much lighter control traffic overhead.

When the VNLayer uses only Server messages from its leader to check for state inconsistencies, the number of state synchronizations is reduced. VNAODV's PDF curve is flatter due to the reduced state synchronization traffic. On top of this, when the VNLayer synchronizes only the hard state part of the routing table, the state synchronization messages are much smaller. With state synchronization traffic greatly reduced, the PDF curve is even flatter. VNAODV now outperforms AODV when the number of CBR sessions is less than 15.

Fig. 3 also shows what happens when state replication among emulator nodes is completely disabled. Without state synchronizations, VNAODV still performs better than the case when the two extensions are not used. This suggests that for VNAODV, keeping routing tables strictly synchronized among emulators is not critical to its performance. At the same time, excessive state synchronizations hurt its performance.

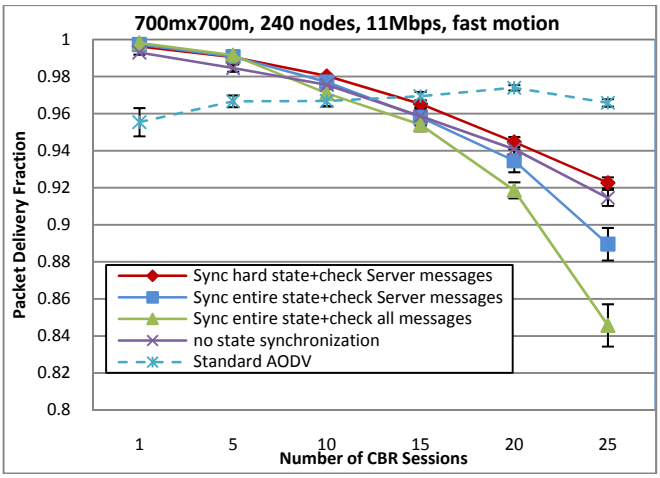

Figure 3. Packet Delivery Fraction of VNAODV with "Selective State Synchronization" and "Selective State Consistency Checks"

B. *Effect of "Long Links" and "Directed Broadcast"*

Fig. 4 shows the impact of Long Links (LL) and Directed Broadcast (DB) on the length of forwarding paths created by VNAODV. Here, all the three extensions mentioned in the last subsection are used by VNAODV in all cases.

Without LL and DB, on average, the forwarding paths created by VNAODV are on average about 2 hops longer than the ones created by AODV. With LL used, the forwarding paths created by VNAODV are only about half a hop longer than the ones created by AODV. Considering the fact that at the first hop, a client process always sends its DMSGs to the local vrouter first, often resulting in an unnecessary forwarding hop, the paths created by VNAODV are already no worse than the ones created by AODV. In our simulations with large networks of 12×12 regions, with the LL extension applied, the lengths of the forwarding paths created by VNAODV are very close to the lengths of the paths created by AODV.

The use of DB increases the average length of forwarding paths a little. This is because the improved data transmission reliability makes long forwarding paths last longer.

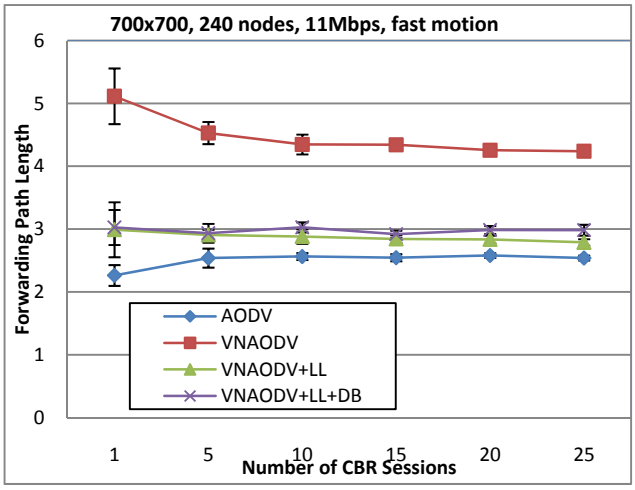

Figure 4. Length of Forwarding Paths of AODV and VNAODV with/without "Long Links" and "Directed Broadcast"

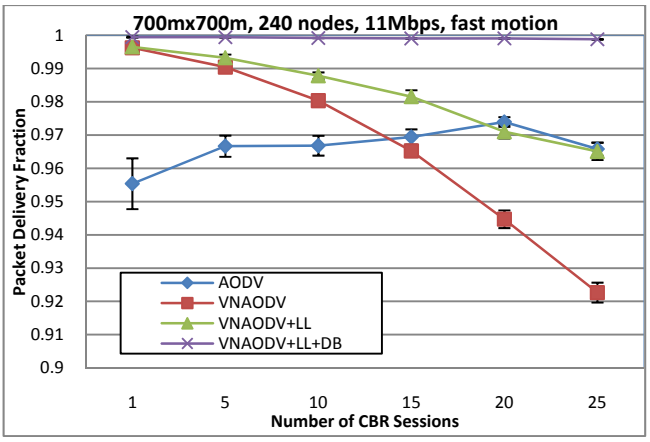

Figure 5. PDF of AODV and VNAODV with/without "Long Links" and "Directed Broadcast"

Shortened forwarding paths lead to shorter forwarding delay, less data forwarding traffic and fewer message losses. Fig. 5 shows the impact of LL and DB on the PDF of VNAODV. We can see that with LL, the PDF of VNAODV is further improved. VNAODV now performs better than AODV when there are less than 20 CBR sessions.

More importantly, Fig. 5 shows that the use of DB drastically improved the PDF of VNAODV and turned the curve almost flat. This is due to the improved reliability on data transmissions between vrouters.

C. *Routing Overhead and Route Stability*

VNAODV's advantage over AODV mainly attribute to the reduced routing overhead. Fig. 6 shows the routing overhead of AODV and VNAODV (with/without LL and DB) with different data traffic loads. Due to the reduced number of routing entities, without LL and DB, VNAODV's routing overhead is already lower than AODV's. LL and DB further reduce VNAODV's routing overhead with shortened forwarding paths and reduced message losses. Fig. 7 shows the total control overhead of AODV and VNAODV, with the VNLayer overhead on state synchronization and leader election taken into account. We can see VNAODV's total control overhead is still lower than the AODV's. Furthermore, it needs to be noted that the VNLayer overhead doesn't cause broadcast storms and is less harmful than the routing overhead.

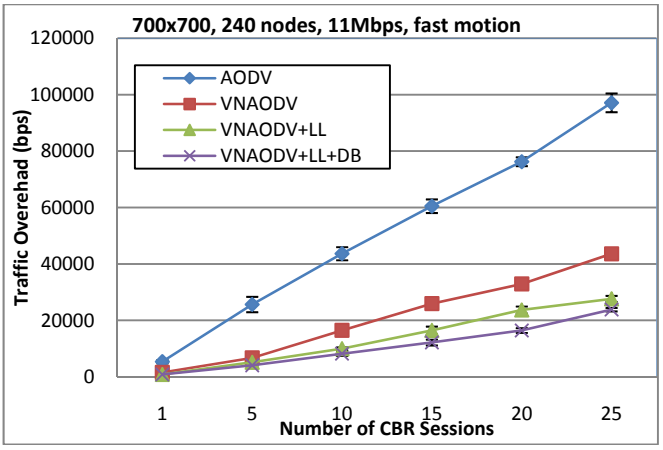

Figure 6. Routing overhead of AODV and VNAODV

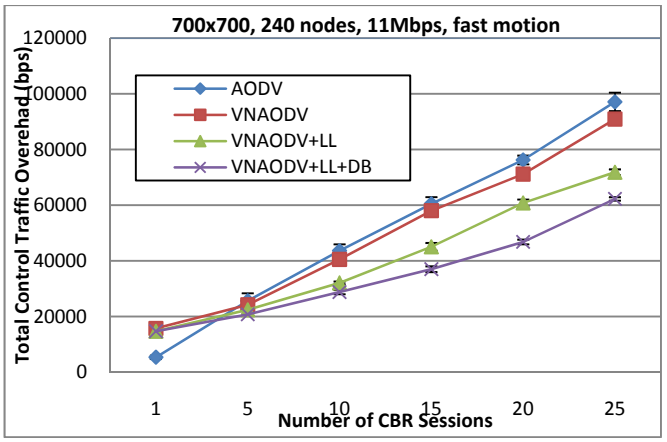

Figure 7. Total control traffic overhead of AODV and VNAODV

We measure route stability by the number of network-wide route discoveries (LCCs not included) and local route repairs done by a protocol in a simulation. Fig. 8 compares the route stability of VNAODV and AODV with different data traffic loads. Here, VNAODV uses all the VNLayer extensions. At every data point, VNAODV does fewer network-wide route discoveries and local route repairs than AODV. This suggests that the forwarding paths created by VNAODV breaks less frequently than the ones created by AODV.

To further verify that the VNLayer approach improves the route stability of VNAODV, we redid the simulations using CBR sessions with sources and destinations that don't move. In this case, the forwarding paths created by VNAODV for the sessions are expected to be more stable than those by AODV. This is because in VNAODV, a forwarding path is a sequence of vrouters. As long as this sequence of vrouters stays up, the forwarding path is good for a session. However, in AODV, the mobility of any router (a physical node) on a forwarding path used by a session can break the forwarding path.

Fig. 8 also shows the route reliability of AODV and VNAODV with CBR sessions using static endpoints. In this case, compared with the corresponding curves for CBR sessions using mobile endpoints, the route stability of AODV and VNAODV both get better. Moreover, as expected, with sessions using static endpoints, VNAODV has a much greater advantage over AODV on route stability.

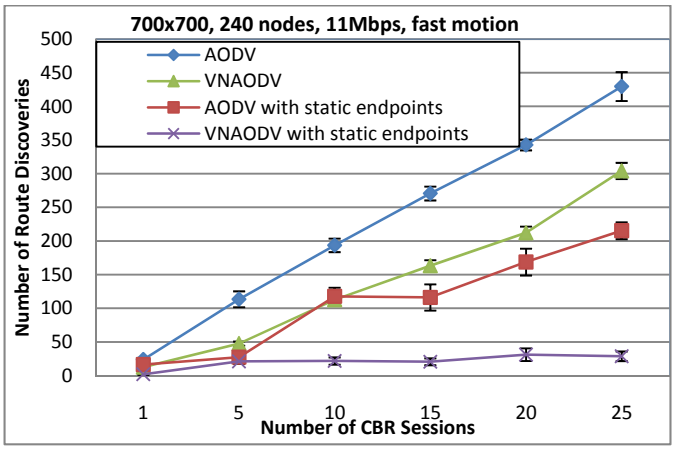

Figure 8. Route Stability of AODV and VNAODV

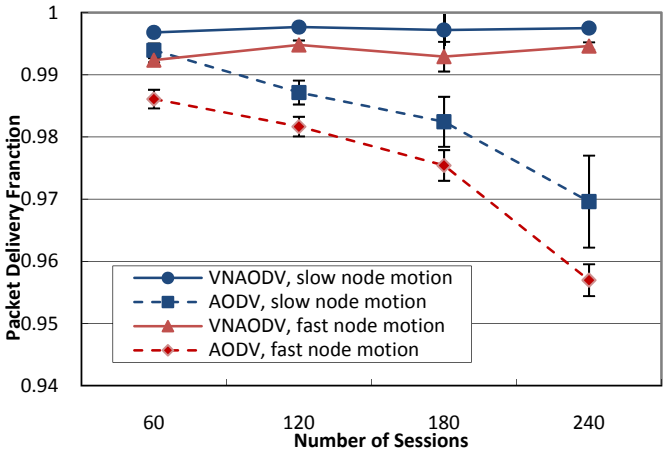

Figure 9. PDF of AODV and VNAODV with different node densities and motion rates

D. *Performance with Different Node Densities and Node Motion Rates*

Fig. 9 compares the PDF of VNAODV and AODV with different node densities and node motion rates (fast mode and slow mode) with 15 CBR sessions. With low network density, VNAODV doesn't have much advantage over AODV. It actually performs a little worse, due to the shortage of backup routers and frequent vrouter failures. However, as the network gets denser, VNAODV scales better than AODV because the number of vrouters is bounded by the number of regions, rather than the total number of physical nodes.

With physical nodes moving faster, as expected, AODV and VNAODV both perform worse. However, Fig. 9 shows that AODV's performance drop with increased node mobility is worse. This suggests that VNAODV is less sensitive to node motion rate increase than AODV, due to the more stable forwarding paths created by VNAODV.

V. Conclusions and Future Work

In this paper, we presented our simulation study on doing reactive MANET routing over the VNLayer. During this study, we identified three major problems with our initial VNLayer implementation. These problems hurt the routing performance badly. To address the problems, we changed the assumptions made by the VNLayer on the link layer and extended the VNLayer implementation.

With the new extensions, the VNLayer is tuned to improve the performance of VNAODV, a traffic intensive application. First, by doing selective state synchronization and selective state consistency checks with incoming messages, the state synchronization traffic was greatly reduced. This leads to fewer message collisions and message losses. Second, by allowing virtual node to communicate across long links, VNAODV's forwarding path length was greatly reduced. Shortened forwarding paths also lead to fewer message delivery failures. Third, the use of Directed Broadcast in data transmissions dramatically reduced message losses, bringing the biggest performance improvement on PDF.

The benefit brought by the extensions come at the cost of weakened state consistency among emulators in a region, reduced reliability of inter-virtual node connections and requirement on the support for promiscuous mode. Since reactive routing protocols are designed to deal with the unreliable, dynamic nature of MANETs, strict state consistency and reliable links are not critical to their performance. For VNAODV, the benefit brought by the extension clearly outweighs the cost.

Since the VNLayer extensions are application independent, they can be used to improve the performance of any application that doesn't demand strict state consistency and needs to deliver a lot of messages across the network. For an application that requires strict state consistency, the first two set of extensions can be turned off.

With the complexity of clustering and state replication handled by the VNLayer, converting AODV into VNAODV is quite easy. This validated that the VNLayer approach can simplify software development for MANET. In addition, simulation results showed that VNAODV outperforms AODV in terms of packet delivery fraction, routing overhead and route stability in a dense network with high node motion rate. This again proved the intuition that the VNLayer approach can improve the efficiency and reliability of MANET protocols.

When there are physical nodes that don't support the promiscuous mode, Directed Broadcast, which had the greatest impact on the PDF of VNAODV, can't be used anymore. Further studies are being done on providing capabilities such as address (virtual node id) resolution, data acknowledgement and retransmission at the VNLayer. With these capabilities, the VNLayer can provide guarantees on data transmissions when they are provided by the link layer.

References

[1] Matthew Brown, Seth Gilbert, Nancy Lynch, Calvin Newport, Tina Nolte, and Michael Spindel. The Virtual Node Layer: A Programming Abstraction for Wireless Sensor Networks. ACM SIGBED Review, 4(3), July 2007.

[2] Mike Spindel, "Simulation and Evaluation of the Reactive Virtual Node Layer", 2007, Master's Thesis at CSAIL, MIT.

[3] Jiang Wu, Nancy Griffeth, Nancy Lynch, Calvin Newport, Ralph Droms, "Simulating Fixed Virtual Nodes for Adapting Wireline Protocols to MANET", The 8th IEEE International Symposium on Network Computing & Applications, Cambridge, MA, July, 2009

[4] C. E. Perkins, E. M. Royer, S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, Network Working Group, IETF

[5] J. Y. Yu and P. Chong, "A survey of clustering schemes for mobile ad hoc networks," in IEEE Communications Surveys & Tutorials, vol. 7, no. 1, First Qtr. 2005.

[6] L. Ritchie, H. S. Yang, A. Richa, and M. Reisslein, "Cluster overlay broadcast (COB): Manet routing with complexity polynomial in source destination distance," Mobile Computing, IEEE Transactions on Publication, vol. 5, no. 6, June, 2006.

[7] R. Sivakumar, P. Sinha, and V. Bharghavan, "Cedar: a core-extraction distributed ad hoc routing algorithm," IEEE Journal on Selected Areas in Communications, 1999, Vol. 17, pages 1454-1465

[8] H. Chen, M. H. T. Martins, P. Huang, H. Cheung So and K. Sezaki. "Cooperative Node Localization for Mobile Sensor Networks. IEEE, 302-308, 2008.

[9] The network simulator - ns-2, http://www.isi.edu/nsnam/ns/